

GNG2101  
**Design Project Progress Update**

**AccessBridge AND 3.4**

Submitted by:

JAMES ATTIA, 300353040

ALEX COTNAM, 300353063

RYAN ATHAUDA, 300374212

ALI GOHAR, 300126709

SAMUEL AGRIPINO DE SOUZA, 300345034

9/18/2024

University of Ottawa

# Table of Contents

---

Project Deliverable Report Instructions .....	i
Table of Contents .....	ii
List of Figures .....	iii
List of Tables .....	iv
List of Acronyms and Glossary .....	v
1 Introduction .....	1
2 Sustainability Report and DFX .....	2
2.1 Sustainability report .....	2
2.2 Design for X .....	2
3 Problem Definition, Concept Development, and Project Plan .....	10
3.1 Problem definition .....	10
3.2 Concept development .....	16
3.3 Project plan .....	22
4 Detailed Design and BOM .....	23
4.1 Detailed design .....	23
4.2 BOM .....	32
4.3 Project plan update .....	33
5 Conclusions .....	34
6 Bibliography .....	35

## List of Figures

---

Insert your list of figures here (right-click to update this field).

List of Tables

---

Table 1. Acronyms.....	v
Table 2. Glossary .....	v

## List of Acronyms and Glossary

---

**Table 1. Acronyms**

Acronym	Definition
LMS	Learning Management System
DFX	Design For Excellence
LCA	Life Cycle Analysis

**Table 2. Glossary**

Term	Acronym	Definition

# 1 Introduction

A Learning Management System (LMS) is a platform used by students and educators to compile and organize course content. Such content should meet certain accessibility constraints, to make shared material more palatable. Rutgers University uses the LMS Canvas, which comes equipped with an accessibility rating tool, called Ally. Unfortunately, the output of an Ally scan can often be daunting to educators, who may not know the best means of improving an accessibility rating.

The goal of this project is to develop an interactive, chatbot-like platform that can recommend directions for improving the accessibility of course content. Ideally, the final product will use the output from Ally to provide sequential steps to improve the most impactful issues first.

This report serves as an outline of the preliminary research and current project plan. The report shall discuss a sustainability report and DFX, problem definition, concept development, the project plan, a detailed design, and a bill of materials.

Before we can create concepts and generate ideas, we need to understand what problem we are attempting to solve. For this, we interpret the client statements into client needs and rank them in terms of importance. Once we have our client needs, we create a problem statement. From the client needs we can also generate metrics to be able to benchmark our product with other competitors. This also allows us generate target specifications. Using the problem definition and the metrics, we can ideate and generate concepts. The concepts are then analyzed against the metrics and client needs to formulate a final global concept. These concepts will be presented to the client to get feedback and preferences on which direction we will take this project.

## 2 Sustainability Report and DFX

### 2.1 Sustainability report

Triple Bottom Line	Positive Impact	Negative Impact
Economic	Time Saving for Educators	Initial Development Cost
	Inexpensive Development	Ongoing Maintenance Cost
	Potential for Funding	Limited Budget for Scaling
Environmental	Reduce Paper Use	Energy Consumption
	Digital Efficiency	Electronic Waste
	Less Physical Distribution	Device Dependency
Social	Support For Students	Learning Curve
	Improves Learning Experience	Reliance on Technology
	Increase Awareness	Support Requirements

#### Economics

- Positive Impact
  - Time saving for educators
    - Our website will make it easier and faster for professors to check and improve the accessibility of their course. Instead of manually reviewing everything, the platform automates the process, saving them time that they can use for other tasks like teaching or preparing a lesson.
  - Inexpensive Development
    - Due to the nature of our product, many facilities, tools and other resources necessary for its development are inexpensive

or free. This means we can build the project at little or no cost, which makes the development phase affordable for us.

- Potential for funding
  - Because our website promotes accessibility, which is an important issue in education, we might be able to get funding or financial support from universities or government programs that focus on improving accessibility for students.

→ Negative Impacts

- Initial development cost
  - Even though we can use free tools, there are still some unavoidable costs, like paying for a website domain, hosting, server hosting, and any other software tools we may need. This could be a financial burden to us, especially if the project requires more resources than expected.
- Ongoing maintenance Cost
  - Once the website is built, it won't run itself. We'll need to update it regularly to keep up with changing accessibility standards and fix any bugs that appear. This means we'll need to spend time and possibly money to maintain the website after it's launched.
- Limited budget for scaling
  - If the website were to become popular and more people start using it, we may need more server space or better technology to handle the traffic. Due to our limited budget, we might struggle to scale up the website to meet this demand without additional funding.

## **Environmental**

→ Positive Impact

- Reduce paper use



- Our website encourages the use of digital materials, which means professors and students won't need to print out as many documents. This helps reduce the amount of paper used, which reduces paper waste and the environmental impact of printing.
- Digital efficiency
  - We plan to design the website to be energy efficient, meaning it will use as little power as possible to run. By writing clean, optimized code, we can minimize the energy needed to operate the site, helping lower the environmental footprint of running it.
- Less physical distribution
  - Since our platform is entirely online, it eliminates the need to physically distribute learning materials, such as shipping books or printing documents. This reduces transportation emissions, which would otherwise contribute to environmental pollution.

→ Negative Impact

- Energy consumption
  - Although our website will be digital, it still requires servers to host it, and these servers need energy to run. If we don't use energy efficient hosting, or servers powered by renewable energy, our website could contribute to higher electrical usage, leading to a negative environmental impact.
- Electronic waste

- Using digital tools means that users need computers, tablets, or smartphones to access our website. Over time, these devices become outdated and get thrown away, adding to the growing issue of technological waste which is harmful to the environment.
- Device dependency
  - Since our platform is fully online, students and professors will need to rely on electronic devices to use it. This means more demand for devices, which requires energy to manufacture, ship, and use leading to a broader environmental footprint due to device dependency.

## **Social**

→ Positive Impact

- Support for students
  - Our website makes it easier for professors to create accessible course materials, which directly benefits students with disabilities. This support ensures that all students, regardless of their ability, have equal access to educational resources.
- Improve learning experiences
  - By making courses more accessible, the overall quality of education improves for everyone. Accessible material tends to be clearer and better organized, which helps all students understand and engage with the content more efficiently.
- Increases awareness
  - Our platform will help raise awareness to the importance of accessibility in education. Professors who may have not been familiar with accessibility needs will learn more about how to make their materials more accessible, promoting a culture to inclusivity awareness to students in needs.

→ Negative Impact

- Learning curve
  - As with any new tool or technology, there will be a learning curve for professors using the website. They may need to spend time understanding how it works and how to implement the accessibility changes, which could be frustrating or time consuming for some.
- Reliance on technology
  - The platform may make professors overly dependent on automated tools for accessibility checks, which could lead to less attention to details. There's a risk that the educators might solely rely on the websites suggestions without fully understanding the underlying accessibility issues.
- Support requirements
  - The website will need ongoing support to help users with questions or technical issues. As a small of individuals, we may not be able to provide long term support after the project's completion which could leave some users struggling to use the platform efficiently over time.

## 2.2 Life Cycle Assessment

Objective:

The primary goal of this LCA is to assess the environmental impact of our accessibility tool throughout its lifecycle. This includes identifying key areas where resource use can be optimized, and environmental effects can be minimized. We will be using metrics from Ally for a real-world comparison.

Scope:

The assessment will include the following phases:

### **Development Phase:**

- Design, coding, and testing of the tool.
- Tools and libraries used for development.

### **Deployment Phase:**

- Hosting infrastructure, including server farms and data centers.
- Ongoing maintenance and updates.

### **Usage Phase:**

- User interactions and the energy footprint associated with data processing and output generation.
- Data transfer during usage.

Inventory Analysis:

### **Development Phase:**

- Resources Used:
  - Development team: 5 developers working for 12 months.
  - Estimated energy consumption for development: 15,000 kWh (based on 100 kWh/month per developer <sup>[1]</sup>).
  - Tools: Cloud-based IDEs and libraries.

### **Deployment Phase:**

- **Hosting Infrastructure:**
  - Cloud hosting service (e.g., AWS, Google Cloud).
  - Estimated monthly energy consumption: 1,500 kWh (considering a PUE of 1.5 <sup>[2]</sup>).
  - Carbon footprint per kWh: 0.4 kg CO<sub>2</sub> <sup>[3]</sup>.

### **Usage Phase:**

- **User Interactions:**
  - Estimated daily document processing: 1,000 documents.
  - Energy consumption per document: 0.1 kWh.
  - Annual interactions: 365,000 documents.
  - Total energy consumption for usage: 36,500 kWh.

### **Impact Assessment:**

#### **Environmental Impact Calculation:**

- **Total Energy Consumption:**
  - Development: 15,000 kWh
  - Deployment: 18,000 kWh (1,500 kWh/month \* 12)
  - Usage: 36,500 kWh
  - **Total Energy:** 69,500 kWh
- **Carbon Emissions:**
  - Total carbon emissions = Total energy consumption \* Carbon emissions per kWh
  - **Calculation:**
    - $69,500 \text{ kWh} * 0.4 \text{ kg CO}_2/\text{kWh} = 27,800 \text{ kg CO}_2$
  - **Annual Carbon Emissions:** Approximately 27.8 metric tons.
- **Resource Efficiency:**
  - Data processed per unit of energy consumed:
    - Total documents processed: 365,000.
    - Total energy for usage: 36,500 kWh.
    - Efficiency:  $365,000 \text{ documents} / 36,500 \text{ kWh} = \sim 10 \text{ documents per kWh}$ .

Interpretation:

The largest contributor to energy consumption and carbon emissions is the usage phase, highlighting the need for optimizing processing efficiency. Development and deployment phases also contribute significantly, indicating potential areas for reducing resource use. To make our product more sustainable and environmentally friendly, we would need to implement energy-efficient coding practices and optimize it in a way to reduce processing power and energy consumption. We can also use hosting servers which use renewable energy for server operations.

## 2.3 Design for X

Design for Reliability

- I. Functions as intended 24/7.
- II. Tool must comply with rules on Web and Mobile accessibility.
- III. Reduces time spent on fixing errors and promotes efficiency.
- IV. Course materials will need to be updated in a timely manner by many different members of Rutgers faculty.

Design for Usability

- I. Used by many members of school faculty of varying experience.
- II. Ease of use promotes efficiency (make desired changes faster).
- III. Should be considered easily usable by >90 percent of users (faculty).
- IV. Improve professor and faculty experience.

Design for Standards

- I. Complies with rules on Web and Mobile accessibility.
- II. Standards are well-defined and universally acceptable.
- III. Following standards ensures that the school meets legal obligations.
- IV. Designing with standards in mind ensures consistency across different courseware.

Design for Simplicity

- I. Simple design minimizes the learning curve for faculty, allowing them to quickly understand and utilize the tool.
- II. Simplicity helps reduce the chances of mistakes during the course creation process.
- III. A straightforward design allows faculty to complete tasks more quickly and efficiently.

- IV. Included in project description based on improvements from previous tool (early design requirement).

#### Design for Testability

- I. Compatible with Canvas platform during testing.
- II. Testing is a major part of the design process to ensure the tool meets requirements.
- III. Easy testing allows for a trial-and-error form of design to work through flaws individually. This will promote efficiency.
- IV. Faster iteration cycles allow for more feedback while meeting deadlines.

### 3 Problem Definition, Concept Development, and Project Plan

#### 3.1 Problem definition

##### Client Statements

#	Client Statements
1	"Often times this ends up being just a long list of red dials, which can be very intimidating (and which leads to people doing nothing)."
2	"Be able to have a tool which can walk you through step by step"
3	"It will come up with a plan for what things to do, that will make the most impact"
4	"Then guide you through those changes with the ability to ask questions as you are on that step before moving on to the next."
5	"Most faculty are not tech savvy and the ability to just have written steps with a means of interaction means

	when there's an issue all work stops."
6	" In the cases of the steps to fix the word doc, you have some cases where the image and where things are may be different, and so the image you see and the instructions you have provided may be slightly different. The example I gave of Mac vs PC is a good example."
7	" It feels like often time what's marketed on a site, is not directly correlated to end user experiences"
8	" I think my ultimate vision is to be able to download the .cvs file which is what we would get on the backend"
9	"It would be nice to be able to upload multiple csv files"
10	"I would prefer if it can be accessed through a web browser, it can be just a local host.
11	" Provides a ChatGPT-like interface to answer questions"

Client Needs



#	Tool	Needs	Importance
1	Accessibility Tool	Output is displayed in an organized fashion.	5
2	Accessibility Tool	Provides instructions in a simple stepwise fashion.	5
3	Accessibility Tool	Prioritize content based on impact of the change.	4
4	Accessibility Tool	Answers questions for a particular step.	3
5	Accessibility Tool	Intuitive/Easy interface.	5
6	Accessibility Tool	Compatible with different operating systems.	4
7	The tool	Functions as advertised.	4

8	Accessibility Tool	Scans content from Ally's csv output file.	5
9	Accessibility Tool	Allows for upload of multiple csv files.	2
10	Accessibility Tool	Accessible through a web browser	5
11	Accessibility Tool	Displays a small chatbot interface	1

What we don't know:

- How much technological knowledge the professors have.
- Exactly what improvements are needed to make the tools easier and less overwhelming.
- The full capabilities of the current Ally tool.
- The contents of the csv file from ally.

Metrics

#	Needs #	Metric	Importance	Units
1	1	How users rate the organization	5	Cumulative Layout Shift Score (No unit)

2	2	How clear the instructions are	5	Fallback Rate (%)
3	3	How well the important changes are shown first	4	Response Accuracy Rate (%)
4	4	How helpful the answers are	3	Fallback Rate (%)
5	5	How easy it is to use	5	Time On Task (Minutes)
6	6	How many different systems it works on	4	Percentage of systems the tool works on (%)
7	7	Does it do what it is said to do	4	Conversion Rate (%)
8	8	How accurately it reads the file	5	Error Detection Rate (%)
9	9	How many csv files it can handle as once	2	Number (number of files)
10	10	How many different web browsers does it work on	5	Number (number of browsers)
11	11	How much space the chatbot uses on the screen	1	Percentage (of space the chatbot uses relative to the screen)

Benchmarking:

#	Units	Yuja Panorama	Blackboard Ally	Ilovepdf	Target Specifications
1	Cumulative Layout Shift Score (No unit)	0.039	0.6	0	<0.1
2	Fallback Rate (%)	N/A	N/A	N/A	<10
3	Response Accuracy Rate (%)	N/A	N/A	99	>85
4	Fallback Rate (%)	N/A	N/A	N/A	<10
5	Time On Task (Minutes)	N/A	N/A	N/A	20
6	Number of operating systems the tool works on (No unit)	3	3	4	>85
7	Conversion Rate (%)	N/A	41-60	100	>75
8	Error Detection Rate (%)	N/A	N/A	N/A	>90
9	Number of files (No unit)	Entire Course	Entire Course	Unspecified	>2
10	Number of supported browsers (No units)	4	4	4	>3
11	Percentage of space the	20-30	20-25	N/A	20-25

	chatbot uses relative to the screen (%)				
--	---	--	--	--	--

## Problem Definition

Develop a webtool that reliably organizes and simplifies the output of Ally into step-by-step intuitive instructions for faculty members.

## 3.2 Concept development

Concept 1:

To create a web tool that can analyze and organize data we will need to use a combination of front and back-end programming languages. For the backend we can use Flask as our web framework, writing in Python. Python excels in handling csv files and the panda libraries give you access to read, analyze and manipulate the data very efficiently. Because flask is a micro-framework, it is very lightweight and minimalist in terms of data size, and it is comparatively easy to use/learn. Having a lightweight web application also means that overall less energy and bandwidth is used which will help with our sustainability goals.

For the front end we can use JavaScript as it is essential for building interactive UIs. We can use React or Django as our main JavaScript library as it allows us to create a simple web page, which can allow users to upload files, trigger analysis and display results. And have additional features like interactive chat bots and step by step instructions, pdf editability, etc.

The way this tool will work is, when you load the application, it will prompt you to login to your canvas account using OAuth2 to be able to access the uploaded files, using canvas public API. Once you log in the tool will then prompt you to upload Ally's report as a CSV file. Once the upload is successful, the app will prompt the user to analyze the report. Once the user clicks analyze the app will analyze and organize all the data and present it to the user. It will display the total number

of accessibility issues and the number of issues that will make the most impact overall. The user can choose either option.

If they choose to view all the issues, it will organize the issues sorted from the most to the least for each file/document. The user can then choose which document they would like to make the changes to. Once the user chooses a document, the tool will then retrieve that document from canvas. The tool will then provide step by step instructions on how to solve each problem, starting from the start of the document to the end. The issues will be displayed in real time, and in certain file formats ex. PDFS the user can edit directly from the tool.

If the user chooses the second option, the tool will organize the documents based on the number of issues that will make the most impact in a descending order. When the user chooses a document to fix, the tool will again provide instructions on how to fix the issues, but only display issues that are either relatively easy to fix or will make a big difference in the accessibility rating of the document. This option will be the intended use of the tool; however, the first option is there for users who may want to fix all the issues in their documents.

Throughout the steps there will be a chatbot icon in the bottom corner which the user can click on at any time. It will have preloaded questions about the current step that may assist less tech savvy users.

## Concept 2:

The website's front end will have a simple, clean, intuitive user interface, coded in JavaScript, consisting of a main box in the center of the screen that displays all text and images associated with solutions, with a small box underneath to upload .csv files and a text box in between to input any inquiries.

The website's back end, which will be coded in python will take supplied .csv file, check the contents against its own database of potential flaws in the content's accessibility and output corresponding solutions to the problems highlighted.

When the user loads up the webtool, they will be prompted to upload their csv file from Ally's output. The tool will analyze and organize the data. It will be presented to the user in two options, fix all issues, or fix by impact rating. If the user selects an option to fix all issues, files will be displayed from the least number of issues to most. If the user selects the latter option, files will be sorted from files with the most impact/easy to fix issues to the least. The user can then select which file they would like to fix. Once selected the tool will prompt the user to open the file on their computer. It will identify the issue by page number, paragraph number, and line number if applicable. It will provide step by step instructions on how to fix the problem. After it has gone through the steps, it will move on to the next issue, and this process will be repeated until all the issues are solved (if user select option 1), or all the impactful changes are made (if user select option 2).

### Concept 3:

For the backend, Node.js and Express will handle server-side tasks. Node.js will manage data processing and user requests, while Express will help create routes, APIs, and other backend functionality. MongoDB will be used as the database to store user data, accessibility reports, and other important information. The app will use OAuth2 to allow users to log in to their Canvas accounts and retrieve uploaded files using Canvas's public API. The backend will also handle file uploads, analyze CSV reports, and efficiently organize the data for display.

On the front end, React will be used to create an interactive and dynamic user interface. Users will be prompted to upload the Ally accessibility report as a CSV file, which they can then analyze. The tool will display the total number of accessibility issues and highlight the ones that will have the biggest impact. Users will have the option to view either all issues or focus on the most important ones. The tool will also provide step-by-step instructions for fixing each issue. For certain formats, like PDFs, users will be able to make edits directly from the tool interface.

Throughout the process, a chatbot will be available at the bottom corner of the screen. The chatbot will offer preloaded questions and answers to assist users, especially those who may not be familiar with accessibility standards or using the tool.

### Concept 4:

The user interface must be simple to use and reliable. It will allow the user to input multiple csv files at one time. Like the other concepts, it will be written in JavaScript. The chatbot interface will allow users to receive real-time feedback. It will feature intuitive prompts guiding users through the accessibility assessment process.

The back end of the web tool will take the csv file and analyze it. It will output suggestions to the user based on a database containing all potential accessibility issues. The back end should return the results to the user, in an ordered list ranked by importance.

### **Global Concept:**

#### **Frontend: React (JavaScript)**

- React will create the user interface, which is the part users see and interact with. This will include buttons, file uploads, and options for viewing the accessibility issues in their documents.
- Users will log in to their Canvas accounts through the app using OAuth2, which is a safe way to access their documents.

- After logging in, users can upload a CSV file, which is a report from Ally that lists all accessibility issues in their documents. The app will then give users two choices: they can either look at all the issues or focus on the ones that will make the biggest difference.
- The app will show the list of documents, with either all issues or the most important ones, depending on what the user selects.
- Step-by-step instructions will guide the user through fixing each issue. If the document is in a format like PDF, users can even make changes directly in the app.
- The app will have a chatbot at the bottom corner. Users can click on it to get help and answers to common questions.

#### Backend: **Node.js** with Express (JavaScript)

- Node.js will handle everything behind the scenes, such as user logins, file uploads, and communication with the Canvas platform.
- Express will manage how the app routes and handles different requests, like uploading files or analyzing the data.
- The app will be able to retrieve documents from Canvas, store data, and figure out which issues in the Ally report are the most important or easiest to fix.

#### Data Processing: **Python**

- Python will process the CSV files that contain accessibility issues. It's really good at reading, analyzing, and sorting through data, making it perfect for this task.
- Once Python analyzes the file, it will send the results back to Node.js, which will display them to the user.

#### Database: **MongoDB (NoSQL)**

- MongoDB will store information, such as user data, accessibility reports, and document details. It can handle lots of different types of data, making it easy to store whatever is needed.
- MongoDB will save user settings and reports so users can come back later and continue their work without starting over.

#### How It **Works**:

1. The user logs in using their Canvas account through OAuth2.
2. The app prompts the user to upload the Ally accessibility report (CSV file).
3. The user can choose to see all issues or focus on the most important ones.



4. The app organizes the issues and gives the user step-by-step instructions for fixing them.
5. A chatbot is always available to help with any questions.

Why Use **MERN** and Python:

- React makes the app fast and interactive for users.
- Node.js and Express help manage everything happening in the background, like handling requests and organizing data.
- Python is great for processing accessibility reports and sorting through data quickly and easily.
- MongoDB stores all the information in a flexible way so the app can grow without any problems.

User **Interface**:

Upon launching the application, users will be prompted to log into their Canvas account via OAuth2 to access uploaded files through the Canvas public API. Once logged in, the tool will prompt users to upload Ally's report as a CSV file. After the upload is complete, the user can proceed to analyze the report. By selecting the "Analyze" option, the tool will process and organize the data, presenting the total number of accessibility issues and highlighting those that will have the most significant impact. Users can choose to view either set of issues.

If the user opts to view all issues, the tool will sort them by severity for each file or document, from most to least significant. The user can then select a specific document to modify, at which point the tool retrieves the document from Canvas. Step-by-step instructions are provided to resolve each issue, working from the beginning of the document to the end. For certain file formats, such as PDFs, users can directly edit within the tool, with issues displayed in real-time.

Alternatively, if the user selects the second option, the tool will sort documents by those with issues that will have the greatest impact, listed in descending order. When the user chooses a document, the tool provides instructions on how to fix only the most impactful or easiest issues to improve the document's accessibility rating. This second option is the tool's primary focus, though the first option is available for those wishing to address all issues within their documents.

Throughout the process, a chatbot icon will be accessible in the bottom corner, offering preloaded questions relevant to the current step, to assist users who may need additional guidance, particularly those less familiar with technology.

To visualize this concept [click here](#).

A few complications may arise with this UI design. The main cause being the unknowns. At this stage, we are not sure how the csv files look, what data/information they provide and how detailed. This concept is designed under the assumption that the csv file from ally gives detailed information on each accessibility issue and its corresponding file. Another complication can arise with the complexity of this concept. We have very strict time constraints, and limited skill/experience with programming. If we run into issues, we will need to pull some features out from the UX and go with a simpler approach as mentioned in concept 2.

### **Relationship with Target Specifications:**

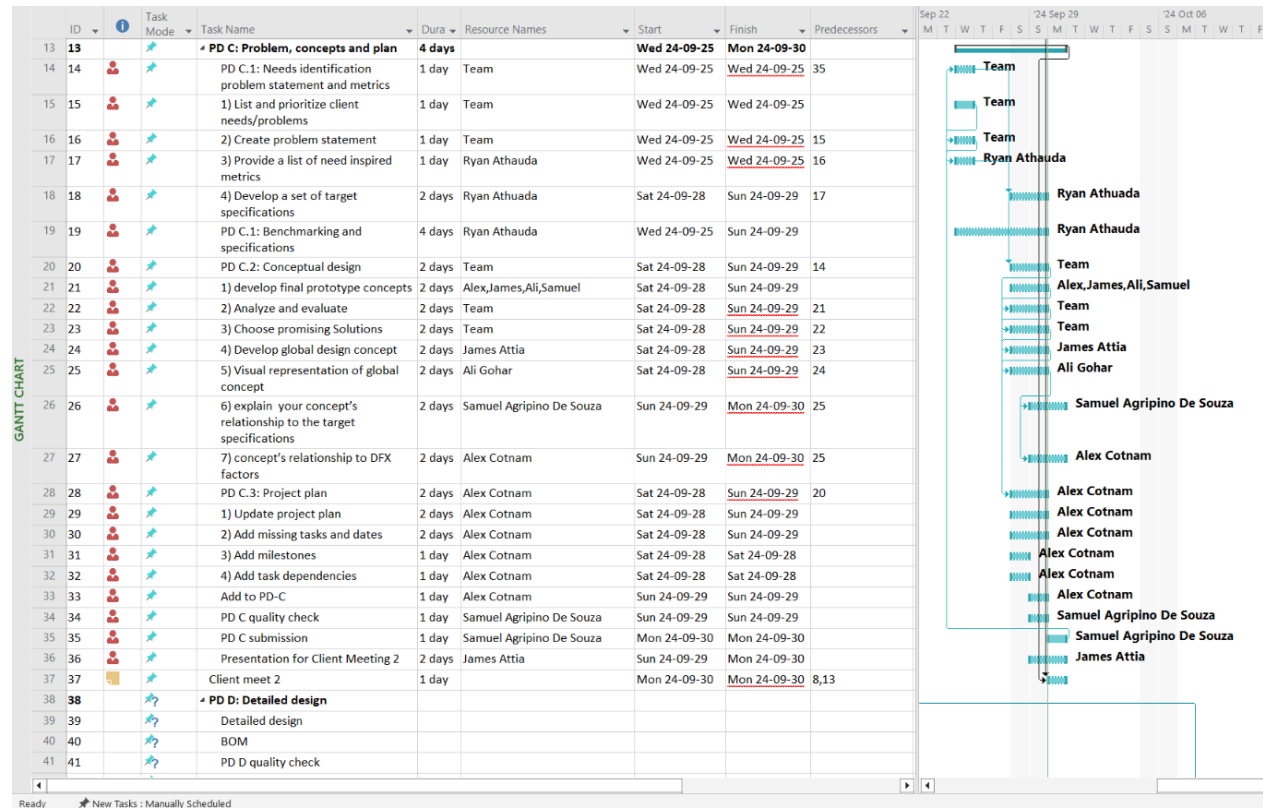
These choices in tools and UI elements are aimed at improving user experience and removing barriers that prevent or slow down the process of making changes to course materials to comply with new accessibility standards. Things like built-in editing tools, chatbot support and simple, intuitive instructions should make it simple for staff to make the necessary modifications without feeling intimidated or overwhelmed. Albeit with some minor inconveniences, such as needing to log in to the website and to drag and drop .csv files into their corresponding input manually, instead of accessing these functions directly from Canvas as the users will be accustomed to.

### **Relationship with DFX Factors:**

The concept relates well to all DFXs listed in PD-B. In terms of reliability, users will be using their OAuth2 credentials to login to Canvas, which as mentioned above, is a safe and reliable way to verify the user's credential. In terms of usability, the entire concept is designed with usability at the forefront. The front end will be simple and non-confusing to allow the user for an efficient and easy-to-use process. For standards, the concept is relatively simple and does not try to reinvent the wheel. Coded in JavaScript and Python, which are well established for the purposes of web tool design. The purpose of standards applies to the tool itself, but also the accessibility standards which the tool is designed to help users fix. Our design will allow users to have the freedom to choose

which changes they would like to make, rather than forcing them to change each issue piece mail and in a specific order. Simplicity relates to usability, as our simple frontend design makes the process simple for any user. Lastly our concept relates to testability because it uses Oauth2. We are provided guest access, as well as a sample csv file which will be crucial for testing, as we can “put ourselves in the user’s shoes” through the entire test process.

### 3.3 Project plan



## 4 Detailed Design and BOM

### 4.1 Detailed design

Our client gave us mostly positive feedback about our design. He particularly likes the global concept and the idea of having seamless integration with canvas using their public Api, and the idea of having live feedback/pdf editable. He did mention that we might need developer access to get access to files for the user. This can be an issue as the concept will need to change some of its functionality if we are unable to get a developer key for Canvas.

We will change the functionality of the program to account for the possibility of not having developer access to canvas. Instead of the app seamlessly downloading each file directly from canvas, it will instead request to upload the file (in case of a pdf file) to allow live changes. For non-PDF files, the program will display the page number and line number where the issue is located.

#### **Application Overview:**

This application provides a platform where users can:

- Sign up or login using Firebase Authentication.
- Upload CSV files from Ally to get a total number of issues, and a total number of critical issues found in the uploaded file.
- Review how many issues each document analyzed by ally contains.
- Upload the desired document and get step by step instructions on how to fix the issues.
- Have live changes and editability for PDF files.
- Save upload history and metrics.

The core technologies used are:

- Firebase: For authentication and Firestore database (not yet used for storage).
- React: For creating interactive components.
- Redux: To manage user state across the app.
- Papa Parse: For handling CSV file parsing and extracting data.

## Functional Breakdown

### Authentication (Login/Signup with Firebase)

#### Subfunctions:

- Login with Email and Password: Uses `signInWithEmailAndPassword` to authenticate users.
- Signup with Email, Password, and Name: Creates a new user with `createUserWithEmailAndPassword` and updates the display name with `updateProfile`.

#### Flow:

- User provides credentials.
- Firebase authenticates (for login) or creates a new user (for signup).
- Redux stores the user's `displayName` for global access.
- On success, the user is redirected to the homepage.
- Error handling provides descriptive feedback if there are issues like incorrect passwords, invalid email formats, etc.

#### Linked Components:

- Homepage (after successful login).
- Redux is used to store user data, which is required across the app.

### File Upload and Data Analysis

#### Subfunctions:

- File Upload: Users can upload CSV files.
- File Parsing and Analysis: CSV data is parsed using Papa Parse, which generates metrics such as:
- Total Issues: Count of issues found in each row.
- Critical Issues: Specific high-priority issues (e.g., missing alt text, poor contrast).

- Overall Score: If available, this calculates the average score for accessibility.

Flow:

- User uploads a CSV file.
- The file is parsed, and data is analyzed.
- The results are displayed immediately in the current session and stored in an array (csvFiles) for viewing in the upload history.
- Each file's history includes:
  - Overall score
  - Total issues
  - Critical issues

Linked Components:

- Upload History Page: Displays all the previously uploaded files with their analysis.
- Redux: Clears user data from global state.

## Header and Sign-Out Functionality

Subfunctions:

- Header: Displays the app logo and a sign-out button.
- Sign-Out: Uses Firebase's signOut function to log the user out, clears the Redux state, and redirects to the login page.

Flow:

- The user can click on "Sign Out," which logs them out from Firebase, clears their session, and brings them back to the login screen.

Linked Components:

- Login/Signup Page: Redirects after sign-out.
- Redux: Clears user data from global state.

## User Interface Design

### Login/Signup Page

- Tabs allow the user to switch between login and signup.
- Each form contains:
  - Login: Email, password fields, and submit button.
  - Signup: Name, email, password fields, and submit button.
- Error messages are displayed in case of invalid input or failed attempts.

### Homepage

- After login, the homepage presents options to upload files or view upload history.

### File Upload Page

- Users can upload CSV files through a drag-and-drop interface or by selecting from their device.
- Once uploaded, a loading bar or spinner could be shown (to simulate analysis time).
- Results are displayed once the file is parsed.

### Document List Page

- Users can view the total issues with each document within the csv file, presented as a list sorted in a ascending order of issues by default.
- Users can sort the documents from a drop-down menu.
- Can select which document to edit.

### Document Edit Page

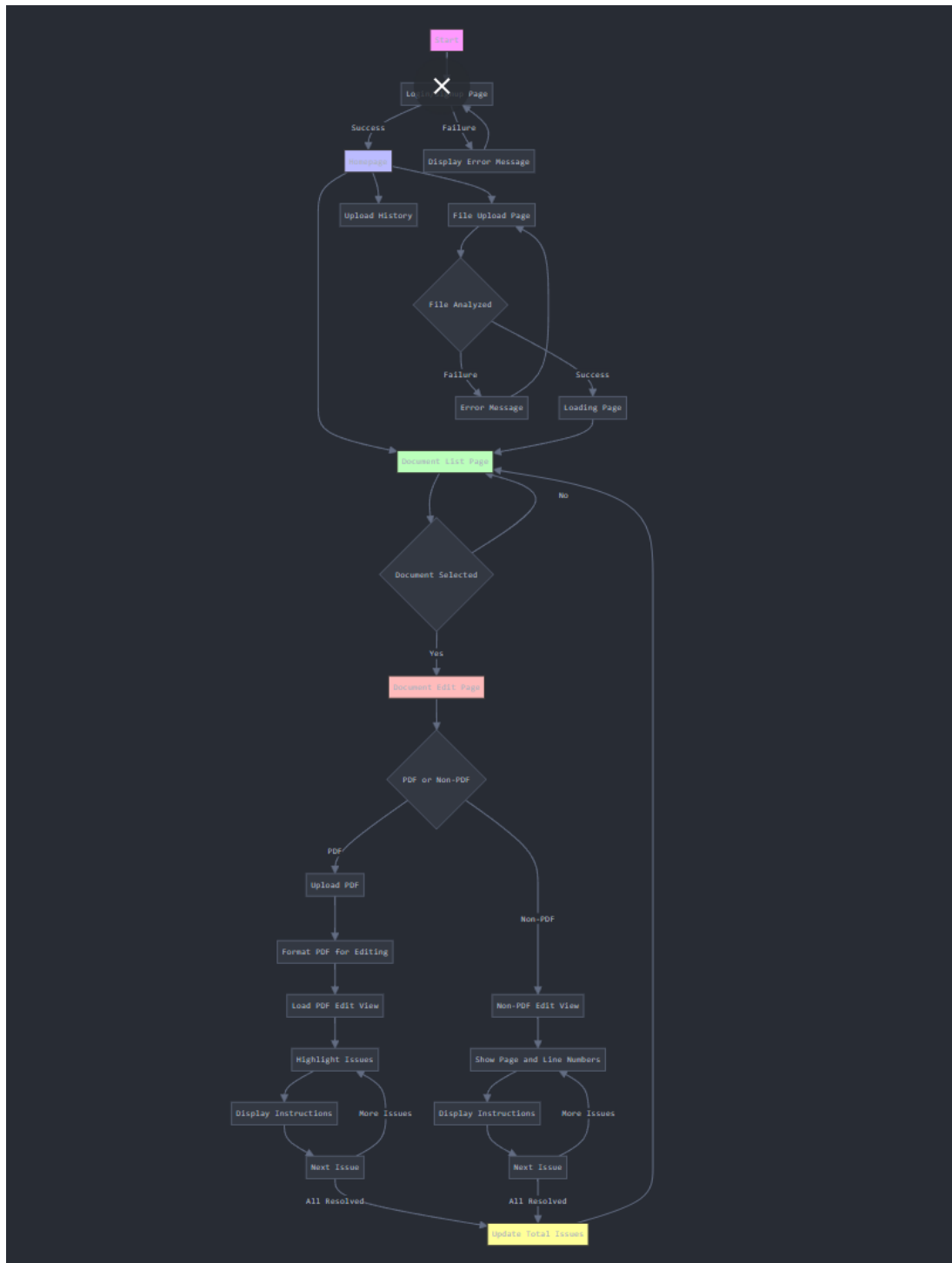
- If the document to be edited is a pdf file:
  - Users can upload the file.
  - Issues highlighted within document, corresponding instructions displayed to fix the issue.
  - Next button can be pressed to move on to the next issue.
  - Users can edit files within the application.
- If non-PDF file:
  - Instructions displayed on how to fix each issue, along with the page number and line number of the issue.
  - No live tracking or changes can be made.

## Upload History Page

- Displays a table of previously uploaded files with:
  - Overall Score
  - Total Issues
  - Critical Issues
- Clicking on a file brings the user back to the detailed analysis results for that file.



## Flowchart



## Detailed Design Considerations

### Design for Excellence (DFX) considerations

When designing or creating our concept to meet DFX goals, there are several important factors we need to consider. These DFX goals include reliability, usability, standards compliance, simplicity, and testability.

#### 1. Design for Reliability

We need to make sure the tool always works reliably, even under different conditions. The tool should comply with web and mobile accessibility rules to ensure it works well for everyone. This will help reduce the time spend fixing problems and make everything run more smoothly. Reliability is crucial because it ensures the tool is trusted and meets user expectations.

#### 2. Design for Usability

The tool should be easy for teachers with different levels of technical skills to use. A simple and easy-to-understand interface makes things more efficient and provides a better user experience. The tool should be accessible to at least 90% of our users, allowing them to make changes quickly. Usability is vital because it determines how effectively people can use the tool and whether they will continue using it.

#### 3. Design for Standards

The tool should comply with well-known web and mobile accessibility standards. This ensures it is both legally compliant and delivers a consistent experience across different platforms. Adhering to standards is important to avoid legal issues and to guarantee that the tool works reliability in various environments.

#### 4. Design for Simplicity

The design should be straightforward to minimize the learning curve for users. A simple design will help teachers understand and use the tool quickly, reducing mistakes and improving efficiency. Simplicity is closely linked to usability, making it a crucial part of our design.

#### 5. Design for Testability

The tool should be easy to test and compatible with the canvas platform. Testability is important to identify and fix any problems during development. This helps us improve the tool more efficiently and meet deadlines. Easy testing also supports trial and error testing to refine the final product.

Overall, while all DFX goals are important, usability, standards compliance, and reliability are the top priority because they directly affect the user experience, legal requirements, and the overall performance of the tool. Simplicity and testability support these main goals by ensuring the tool is easy to use and maintain.

#### 4. List of Skills and Resources

Name	Skill or Resource	Description
Redux	Resource	used to store user data
FireBase	Resource	For authentication and Firestore database
Papa Parse	Resource	For handling CSV file parsing and extracting data
React	Resource	For creating interactive components
Back End Design	Skill	Implementation of various back-end software. Relies on programming
Front End Design	Skill	User Interface. Overall aesthetics and website layout.

#### 5. Timeline Assessment:

At this point, the team is approximately 4 weeks into the project. So far, project planning, user and client needs, and identifying key resources has been established. A global concept has been developed, and we are in the early stages of design. The remaining tasks include front end and

backend development, prototyping, and testing. Front-end development should be done relatively quickly. Back-end development could prove to be difficult. The design has several components that must interact together seamlessly and interact with the front-end interface without issue. We can estimate that the first prototype will take around 1-2 weeks to design, while the total prototyping stage could take 4-6 weeks. Testing should take a few days to one week once the final prototype has been drafted. By this time, Design Day will be 1-2 weeks away. Testing will continue until design day to ensure that the “final product” is presentable, functional, and meets standards defined previously.

The team meets for 3 hours during Monday Lab sessions, 2 hours every Wednesday afternoon, and during lectures (80 minutes x 2/week). During these times, project work is not always a priority. During meetings, we typically discuss the project as well as deliverables and assign tasks for both. Most project work thus far is typically done outside of these times. During weekdays, all team members are pretty busy with other obligations. Weekends tend to be the best time to complete project work. We can estimate that each team member has around 4 to 8 hours of time per week to complete project related tasks, varying depending on individual exam schedules and other obligations. This number may need to increase or decrease, which we will continue to gauge throughout the coming weeks.

## **6. Critical Product Assumptions**

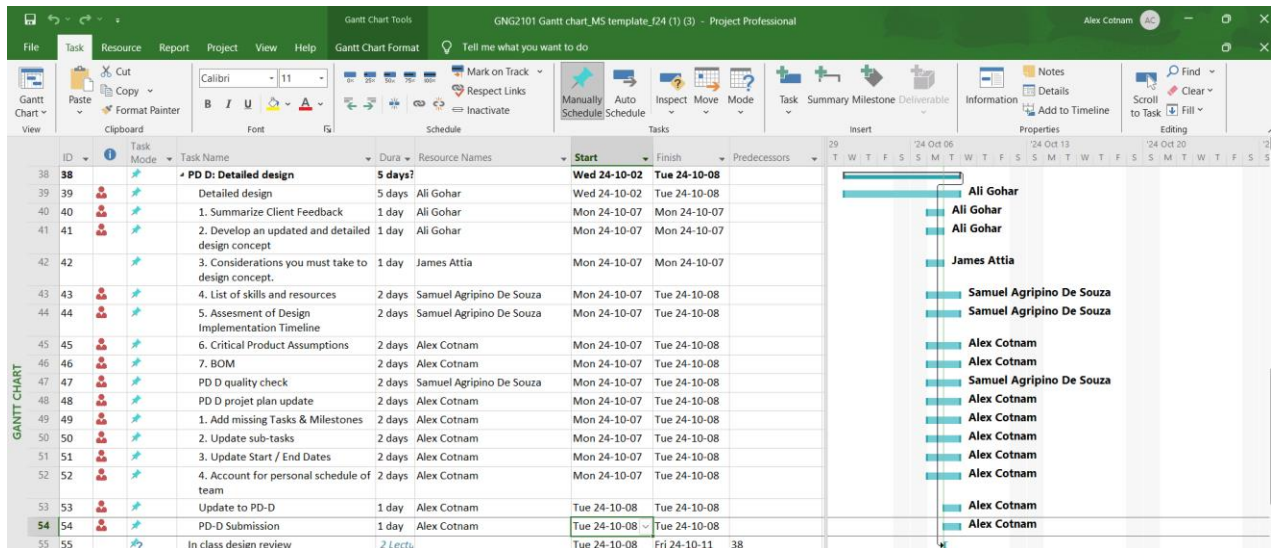
In the case of our project, physical materials are not a concern. We do not need to go purchase or order anything, print anything etc. The items listed in the Bill of Materials have already

been implemented on our devices. As far as what we need from our client, we have guest access to a Rutgers.edu account, as well as an example csv file for our use. The values listed under the target specifications table will be satisfied during testing throughout the design process. This could affect the ability to implement the design, for example, if the error detection rate falls below 90%, changes will have to be made, possibly affecting the project schedule. The implementation of some of our software is new to at least a few of the team members. This always has the potential for issues to arise, particularly with the back end of the web application.

## 8.2 BOM

Item No.	Name of application	Description	Cost (CAD)	Quantity	Total Cost (CAD)
1	Rutgers NetID	Guest access account to Rutgers Canvas platform	\$0.00	1	
2	Visual Studio Code	Code editor for web applications, used as an IDE	\$0.00	5	
3	Node.js	JavaScript runtime environment	\$0.00	5	
4	Git	DevOps tool used for source code management.	\$0.00	5	
5	FireBase	Used for account authentication & storage of account IDs	\$0.00	1	
					\$0.00

# 4.3 Project plan update



## 5 Conclusions

Given our analysis, our product has the potential to reduce costs for universities seeking to improve the accessibility of their courses, due to the improvements in efficiency caused by the time savings afforded by our product. This increase in efficiency will allow professors to improve the accessibility of their course material to more easily, enhancing the learning experience for students of varying need and ability.

Moreover, the low required cost of development due to the nature of our tool and the potential for future funding from educational institutions make this project much easier to pursue. Our application would also increase the relevance of digital material, which has a positive impact in reducing reliance on physical materials but may conversely increase dependence on digital tools. Additionally, for the time being our budget is relatively low, and any small increase in development or maintenance costs will have a great impact.

To maximize the positives and minimize the negatives outlined above, our life cycle analysis highlights key points to improve upon our tool's effectiveness and efficiency.

For our product to be successful it must function reliably, have a simple and intuitive user interface that allows for great useability, properly convey the accessibility standards outlined in the rules on Web and Mobile accessibility and be easily testable for improved efficiency in development and maintenance.

Our client's statements and our research/observations alike have heavily influenced the specifications we are targeting for performance and useability, as well as the general concept we intend to move forward with, after we receive feedback on our progress. The choices we made in user interface layout and programming architecture were made to push us closer to meeting the needs and wants of both our client and the end users of our product.

PD-D focuses on the specifics of a refined and detailed design following client meeting 2 and in preparation for the In-Class design review. Client feedback was very positive, particularly with regard to our Global Concept. A realistic assessment of the time required for design implementation, including prototyping and testing was also listed. The potential for problems to arise with respect to acceptable values of a specification was overviewed. Considerations regarding certain issues, such as the physical availability of a product was deemed unimportant. This can be considered an advantage when compared to other teams with physical product projects. Resources and skills at the disposal of the team have also been highlighted here, as well as major design considerations and their relation to DFX factors. Finally, a Bill of Materials table was created, detailing the cost, quantity, and description of resources required by the team. The Project Skeleton was also updated to include all subtasks and their owner for this deliverable, as well as their respective dates.

## 6 Bibliography

- [1] U.S. Environmental Protection Agency ENERGY STAR Program. (2007, Aug 2). *Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431*. Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431 | Energystar.gov. <https://www.energystar.gov/buildings/tools-and-resources/report-congress-server-and-data-center-energy-efficiency>
- [2] Shehabi, A., Smith, S., Sartor, D., Brown, R., Herrlin, M., Koomey, J., Masanet, E., Horner, N., Azevedo, I., & Lintner, W. (2016, June 1). *United States Data Center Energy Usage Report*. United States Data Center Energy Usage Report (Technical Report) | OSTI.GOV. <https://www.osti.gov/biblio/1372902>
- [3] *How much carbon dioxide is produced per kilowatthour of U.S. electricity generation?* - U.S. energy information administration (EIA). Frequently Asked Questions (FAQs) - U.S. Energy Information Administration (EIA). (n.d.). <https://www.eia.gov/tools/faqs/faq.php?id=74&t=11>
- [4] Touré, H. I. (n.d.). *ITU and Climate Change Report*. ITU and climate change. <https://www.itu.int/themes/climate/docs/report/index.html>